We Claim:

1. A method for generating an application specific program utilizing an abbreviated instruction set comprising the steps of:

generating a native program for an application utilizing a set of native instructions;

debugging the native program;

processing the debugged native program to determine an abbreviated instruction set corresponding to the set of native instructions; and

converting the native program to the application specific program by replacing the set of native instructions with the abbreviated instruction set.

2. The method of claim 1 wherein said step of processing further comprises:

analyzing the set of native instructions to identify a first group of native instructions having a style pattern of bits which is defined as a specific pattern of bits that are constant for said group.

3. The method of claim 2 further comprising the step of:

storing the identified style pattern of bits in a translation memory.

4. The method of claim 3 further comprising the step of:

utilizing the identified style pattern of bits stored in said translation memory to recreate native instructions from the first group of native instruction by combining bits from corresponding abbreviated instructions with the identified style pattern of bits.

5. The method of claim 1 wherein said step of processing further comprises:

analyzing the set of native instructions to identify multiple groups of native instructions, each group having a style pattern of bits which is defined as a specific pattern of bits that are constant.

6. The method of claim 5 further comprising the step of:

storing the identified style patterns of bits in a translation memory.

7. The method of claim 6 further comprising the step of:

utilizing an identified style pattern of bits selected from said translation memory to recreate native instructions from one of said multiple groups of native instructions by combining bits from corresponding abbreviated instructions with the identified style pattern of bits.

8. The method of claim 4 further comprising the step of:

creating a one-to-one mapping between a program's native instruction and an abbreviated instruction by using a translation memory addressing mechanism to identify the style pattern of bits stored in said translation memory.

9. The method of claim 7 further comprising the step of:

creating a one-to-one mapping between a program's native instruction and an abbreviated instruction by using a translation memory addressing mechanism to identify the style pattern of bits stored in said translation memory.

10. The method of claim 8 further comprising the translation memory addressing step of adding or concatenating an offset field contained in the abbreviated instruction with a translation memory base address stored in an internal machine register to form the address to select a specific pattern of bits from said translation memory.

11. The method of claim 9 further comprising the translation memory addressing step of adding or concatenating an offset field contained in the abbreviated instruction with a translation memory base address stored in an internal machine register to form the address to select a specific pattern of bits from said translation memory.

12. The method of claim 1 further comprising the step of:

executing the application specific program on a simulator to verify its functional equivalence to the native program.

13. The method of claim 12 further comprising the step of:

determining a processor core specification tailored for use in implementing the application specific program utilizing the abbreviated instruction set.

14. The method of claim 1 further comprising the step of executing the application specific program on a Manta-2 based simulator acting as an emulator.

15. The method of claim 1 wherein the native instruction set is a manifold array (ManArray) instruction set.

16. The method of claim 15 further comprising the step of translating abbreviated instructions back into a native ManArray format for decoding and execution in a ManArray sequence processor and processing elements.

17. The method of claim 16 wherein the step of translating abbreviated instructions back is performed only by a sequence processor.

18. A method for generating an abbreviated instruction set corresponding to a set of native manifold array (ManArray) instructions for an application specific program comprising the steps of:

separating the set of native ManArray instructions into groups of instructions;

identifying the unique instructions within each group of instructions;

analyzing the native instructions for common instruction characteristics;

determining at least one style pattern of bits which is defined as a specific pattern of bits that are constant; and

generating the abbreviated instruction set utilizing the at least one style.

19. The method of claim 18 wherein the set of native ManArray instructions are separated into groups by classifying said instructions by opcode.

20. The method of claim 19 wherein at least the following groups are established: store and load instructions; MAU and ALU instructions; DSU instructions; and control flow instructions.

21. The method of claim 19 wherein at least one of the common instruction characteristics is a relative bit-pattern usage in the application specific program for a given bit-pattern split in an abbreviated instruction format.

22. The method of claim 18 further comprising the step of:

storing the at least one style pattern of bits in a translation memory.

23. The method of claim 22 further comprising the step of analyzing relative bit-pattern usage among groups of instructions that include a common style.

24. The method of claim 22 further comprising the step of generating at least one translation management memory instruction.

25. The method of claim 22 further comprising the step of:

utilizing the identified style stored in the translation memory to recreate native instructions from a first group of native instruction by combining bits from corresponding abbreviated instructions with the at least one style pattern of bits.

26. A method for translating abbreviated instructions into a native instruction format comprising the steps of:

fetching an abbreviated instruction from a memory tailored to storage of abbreviated instructions;

dynamically translating the abbreviated instruction into the format of a native instruction in a sequence processor (SP) array controller; and

dispatching the native instruction to a processing element for execution.

27. The method of claim 26 wherein the abbreviated instruction includes at least one S/P bit, a multi-bit opcode field and a multi-bit translation memory address offset.

28. The method of claim 27 wherein the step of dynamically translating further comprises the step of decoding the multi-bit opcode field.

29. The method of claim 27 wherein the step of dynamically translating further comprises the steps of forming a translation memory address by adding the multi-bit translation memory address offset with a translation memory base address; and

selecting a plurality of native instruction bits from a location in the translation memory corresponding to the formed translation memory address.

30. The method of claim 27 further comprising the step of directly using the multi-bit translation memory address offset to select a plurality of native instruction bits from a location in a translation memory corresponding to the multi-bit translation memory address offset.

31. The method of claim 26 wherein the abbreviated instruction includes at least one S/P bit, a multi-bit opcode field, a multi-bit translation memory address offset, and a plurality of bits which are to be directly loaded.

32. The method of claim 31 wherein the step of dynamically translating further comprises the step of decoding the multi-bit opcode field.

33. The method of claim 31 wherein the step of dynamically translating further comprises the steps of forming a translation memory address by adding the multi-bit translation memory offset with a translation memory base address; and

selecting a plurality of native instruction bits from a location in the translation memory corresponding to the formed translation memory address.

34. The method of claim 33 wherein the step of dynamically translating further comprises the step of combining the selected plurality of native instruction bits and the plurality of bits which are to be directly entered.

35. The method of claim 26 wherein the abbreviated instruction includes at least one S/P bit, a multi-bit opcode field, a first multi-bit translation memory offset field and a second multi-bit translation memory offset field.

36. The method of claim 35 wherein the step of dynamically translating further comprises the step of decoding the multi-bit opcode field.

37. The method of claim 35 wherein the step of dynamically translating further comprises the steps of:

selecting a first multi-bit portion of the native instruction from a first translation memory address utilizing the first multi-bit translation memory offset field; and

selecting a second multi-bit portion of the native instruction from a second translation memory address utilizing the second multi-bit translation memory offset field; and

combining both multi-bit portions into a native instruction format.

38. The method of claim 37 wherein translation memory addresses are formed by concatenating base address register bits and translation memory offset field bits.

39. A system for controlling a translation process wherein a B-bit abbreviated instruction is translated into a native instruction format including a number of bits C greater than B, the system comprising:

a B-bit instruction register;

a base register;

an adder;

a decoder;

a translation memory; and

a native instruction register, wherein the base register output and a field of the B-bit abbreviated instruction in the B-bit instruction register are added by the adder to produce an output which selects native instruction bits in the translation memory for loading into the native instruction register.

40. The system of claim 39 wherein the decoder receives opcode bits from the B-bit abbreviated instruction in the B-bit instruction register and decodes said opcode bits to generate group bits which are loaded into the native instruction register.

41. The system of claim 39 wherein B is 12, 13, 14, 15, 16, or some other integer value less than 30 and C is 32, 40, 48 or 64.

42. A system for controlling a translation process wherein a B-bit abbreviated instruction is translated into a native instruction format including a number of bits C greater than B, the system comprising:

a B-bit instruction register;

a base register;

an adder;

a decoder;

a translation memory; and

a native instruction register, wherein the native instruction register receives a plurality of direct load bits from a direct load field of the abbreviated instruction in the B-bit instruction register; and

a base register output and a field of the B-bit abbreviated instruction are added by the adder to produce an output which selects native instruction bits in the translation memory for loading in combination with the direct load bits into the native instruction register.

43. A system for controlling a translation process wherein a B-bit abbreviated instruction is translated into a native instruction format including a number of bits C greater than B, the system comprising:

a B-bit instruction register;

two base registers;

a decoder;

two translation memories; and

a native instruction register, wherein the native instruction register receives a plurality of translation bits from both translation memories that are combined as specified by a style set of bits stored in the processor; and two base register outputs and two fields of the B-bit abbreviated instruction which are concatenated respectively to form two translation memory addresses to select native instruction bits in the translation memory for loading into the native instruction register.

44. A process for executing a multiple stage pipeline utilizing abbreviated instructions and an expand and dispatch stage, the process comprising the steps of:

fetching a first abbreviated B-bit instruction over an instruction bus from a reduced size instruction memory during a first fetch cycle;

loading the fetched abbreviated B-bit instruction into a first instruction register;

fetching a second abbreviated instruction in a second fetch cycle;

operating on the first abbreviated B-bit instruction in the expand and dispatch stage during the second fetch cycle so as to load a native form of the abbreviated B-bit instruction into a second instruction register at the end of the second fetch cycle;

loading the second abbreviated instruction into the first instruction register;

fetching a third abbreviated B-bit instruction in a third fetch cycle; and

operating on the second abbreviated instruction during the third fetch cycle so as to load a native form of the second abbreviated instruction into the second instruction register at the end of the third fetch cycle.

45. The process of claim 44 wherein a fetched abbreviated XV indirect execute VLIW instruction associated very long instruction word instruction memory (VIM) address is calculated by an address generation function in the expand and dispatch pipeline stage in parallel with the translation of the abbreviated XV instruction.

46. The process of claim 44 wherein the second fetched abbreviated instruction is an abbreviated XV instruction.

47. The process of claim 46 wherein during a fourth fetch cycle:

a fourth abbreviated B-bit instruction is fetched and loaded;

the third abbreviated B-bit instruction is translated into native format;

the very long instruction word fetched from VIM address is in a decoder; and

the native form of the first abbreviated B-bit instruction has entered an execute pipeline stage.

48. The process of claim 47 wherein during a fifth fetch cycle:

a fifth abbreviated B-bit instruction is fetched and loaded;

the fourth abbreviated B-bit instruction is translated into native format;

the native form of the third abbreviated B-bit instruction is decoded; and

the fetched very long instruction word enters the execute stage of the pipeline.

49. A system for translating abbreviated instructions into a native instruction format comprising:

a memory storing an abbreviated instruction;

means for fetching the abbreviated instruction from the memory; and

means for dynamically translating the abbreviated instruction into a native instruction in the native instruction format.

50. The system of claim 49 further comprising means for dispatching the native instruction to at least one processing element for execution.

51. The system of claim 49 wherein the means for dynamically translating further comprises at least one translation memory for storing style pattern bits which are common to a group of native instructions.

52. A dual fetch processing system employing a dual abbreviated instruction format for reduced power operation of a core processor, the system comprising:

an abbreviated instruction memory storing ((j/2) + k) x B-bit instructions corresponding to j x C-bit native application program instructions, where B is less than C and k represents the number of translation memory base register management instructions;

an abbreviated instruction register;

an abbreviated instruction pre-register;

means for fetching two abbreviated instructions at a time and loading one into the abbreviated instruction pre-register and one into the abbreviated instruction register; and

means for processing the abbreviated instruction in the abbreviated instruction register.

53. The system of claim 52 further comprising a B-bit instruction bus which is split into two segments, a first segment connecting the abbreviated instruction memory through a controllable switch to the abbreviated instruction register and a second segment connecting the abbreviated instruction memory to the abbreviated instruction pre-register.

54. The system of claim 52 wherein the first and second segments are unequal.

55. The system of claim 52 further comprising:

first and second base address registers; and

first and second translation memories.

56. The system of claim 52 further comprising:

an instruction flow control unit; and

an iVLIW memory.

ADD B5